RT-MaG ToolBox
User's guide

Augustin Manecy

March 21, 2014

Contents

1 Description

This toolbox consists of a rapid prototyping tool chain for autonomous robotic applications using Matlab Simulink. This toolbox allows to program and monitor directly a Computer On Module (COM) through Matlab Simulink. RT-MaG toolbox frees the users from writing low level language code. RT-MaG toolbox gives direct access to the I/Os through intuitive masked Simulink blocks. Finally, the user just needs to design its Simulink model with the wished I/Os, as done for example with dSPACE, or MPLAB 16-Bit Device Blocks for Simulink. Then a simple call to a function allows to generate the complete standalone real-time application of this Simulink model. To summarize the RTMAG toolbox allows to:

- automatically generate a multi-rate real-time application from a Simulink model for a real-time embedded Linux environment,
- make the developments really easy, fast and bug free,
- develop the Gumstix Simulink model directly in 32-bits floating point,
- monitor, log and plot data sent by the COM in real-time via a Serial or a Wifi connection,
- design applications fully compatible with all Simulink blocks (embedded matlab function, s-function, matlab data-structure...),
- send in real time high level set-points for the controllers embedded on the COM,
- tune in real time controller's gains and parameters thanks to a Simulink host computer's application,
- have the possibility to start and stop the application at any time (thanks to the wireless connection),
- monitor continuously the CPU load, overruns and the execution time of each task,
- have an access to a "debug mode": detailed execution time of each task, time of I/Os access, and a clear message for I/Os errors,
- test programs both in Processor-In-the-Loop (PIL) and RunTime mode

The toolbox will configure automatically the COM and you can be concentrated only on the control of your robot.

1.1 Application structure

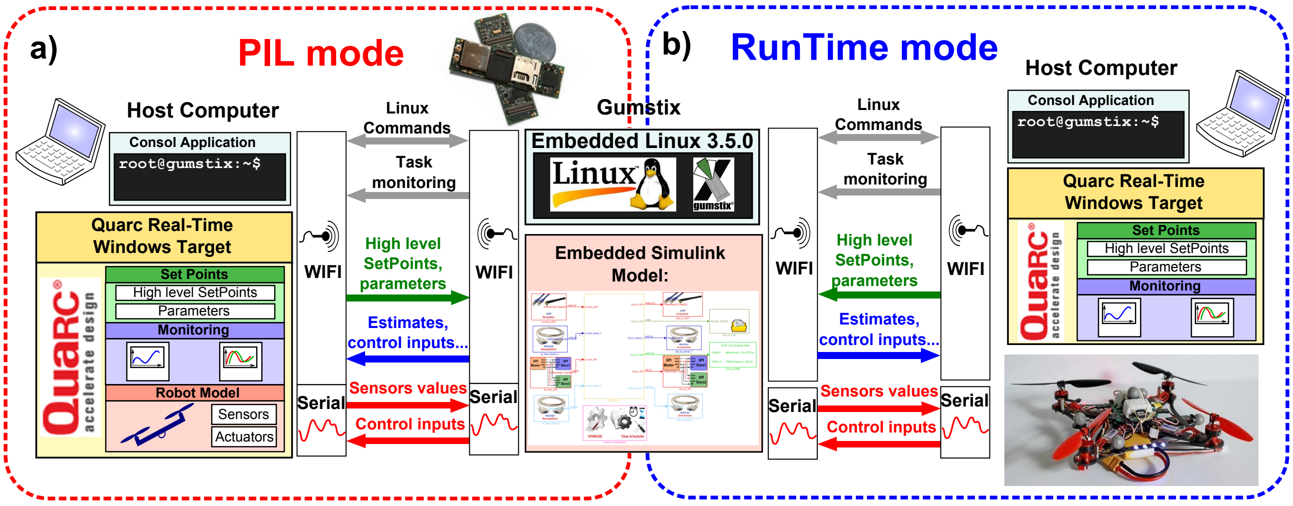


Figure 1: a) The Processor-In-the-Loop mode allows to validate and test the robotic application in real-time on the COM, without damaging the robot. Here the PIL was used to validate the controllers of the X4-MaG Quadrotor. The host computer could be used to emulate the RS232 connection of a low level controller and to simulate the robot behaviour. b) The RunTime mode, i.e. when the robot is running. This application has the same interfaces than in the PIL mode. As it could be seen, the application of the COM remains the same which avoid new bug to appear.

The RT-MaG applications are composed of two parts:

- a real-time embedded application which executes in real-time a Simulink model on a Computer-On-Module (COM) to control a robot,
- a real-time host application which executes in real-time a Simulink model on a host computer. The host computer is used not only to send high level setpoints and parameters to the embedded application but also to monitor all the signals the users want to check.

The COM application could be monitored by two ways:

- a ssh connection via a console application (e.g., PUTTY) to monitor the CPU load, the eventual overruns, I/Os error, etc.
- the host Simulink application to monitor the computations.

1.2 Toolbox's workflow

RT-MaG generates then the two real-time applications, the steps are described by the figure 2:

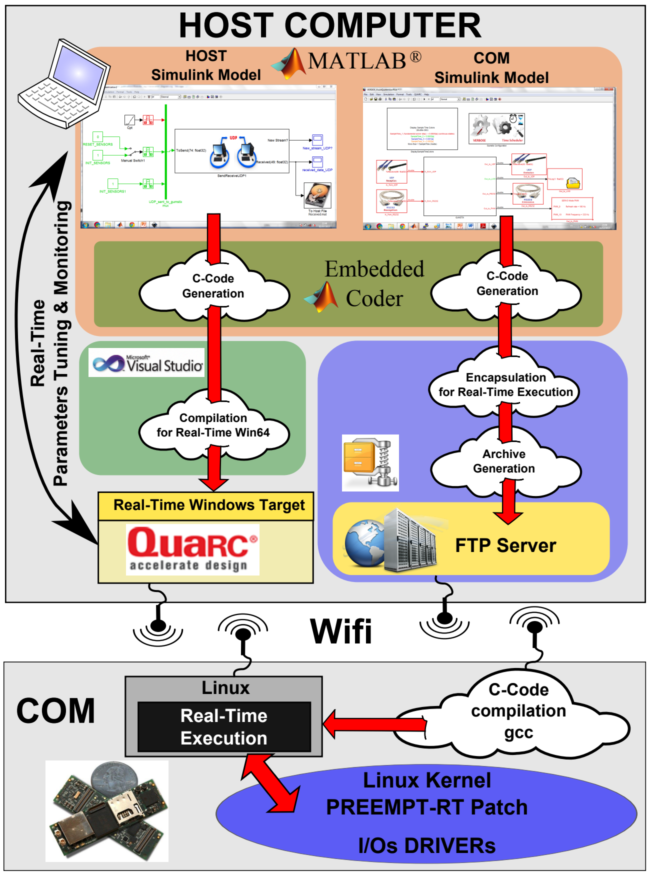


Figure 2: WorkFlow to generate the real-time application, all these steps are done almost automatically.

Figure 2 describes the different steps performed by the toolbox to obtain a real-time application. When the Simulink model of the COM is completed, the toolbox generates automatically an archive containing all the files which are required to be compiled. The host real-time application is also generated as an external application which could be tuned and monitor in real-time thanks to QUARC. Once the application have been generated their is two different part:

- An archive (.zip) containing all the files needed to build the standalone application on the COM. This archive was automatically copied to the FTP server before to be downloaded by the COM to be build,
- A windows real-time Simulink application (.rt-win64) which will be executed by an external real-time target (QUARC) connected to the Simulink model. In this Simulink model it is possible to visualize in real-time the signals received from the COM by an UDP connection and to send parameters to the COM. This model could be used to simulate a robot behaviour in a PIL context.

2 Starter guide

2.1 Requirements

To follow this tutorial, you have to prepare your COM to run RT-MaG programs. Please refer to the download page of our website to flash your COM with one of Linux images if your COM is supported. Or refer to RT-MaG_ToolBox_Prepare_your_COM.pdf guide to make your COM ready to run RT-MaG programs if their is no images available for your COM.

2.2 RT-MaG tools for your COM

2.2.1 Development computer tools

Here you can find a description of what is present in the toolbox archive. Once the archive of the toolbox was inflated and installed, you can explorer the different directories:

- **README.txt**: a file with some basic instructions to install RT-MaG and a description of the current version.
- **Install_RT_MaG_ToolBox*.***: the installation files of the toolbox. To install the toolbox, run Matlab with administrator privileges and execute **Install_RT_MaG_ToolBox.m**.
- *RT_MaG_ToolBox*: all the files of the toolbox containing all the functions, library blocks, etc.
- *Projects_COM*: A directory containing all the demonstration models. You can find here:
  - *Test_Gumstix*: A directory containing al lot of different models to validate that the toolbox is correctly installed.
    You can generate all these model for your COM by running the **TEST_COM_IO_BLOCKs.m**
  - *Test_HIL*: An example of an PIL project containing the host and the COM model.
    You can generate these two model for execution by running **init_COM_Test_HIL.m** and **init_QUARC_Test_HIL.m**
  - *Test_RunTime*: *Test_HIL*: An example of RunTime project containing the host and the COM model.
    You can generate these two model for execution by running **init_COM_Test_RunTime.m** and **init_QUARC_Test_RunTime.m**

Remark 1: Note that all the Matlab functions and scripts of the toolbox provide an help menu. To access to the help, you can just type `>> help FunctionName` in your Matlab command window.

2.2.2 COM tools

The RT-MaG toolbox offers to you some shell scripts allowing to automatize different step of the application generation. You can find here a small description of each script:

- `UpToDateModel <ModelName>`: This script downloads the model called `<ModelName>` and build it automatically in your COM.
- `RunModel <ModelName> -d <Duration>`: This script executes the model `<ModelName>` on your COM for a duration of `<Duration>`.
- `CompileTestModel`: This script downloads automatically and compile all the test models available in the folder ***your_RT-MaG_Folder****/Projects_COM/Test_Gumstix*. You just have to generate the archives before in your development computer.
- `PutFTP`: A simple script to put a file from your COM to the FTP server available on your development computer.

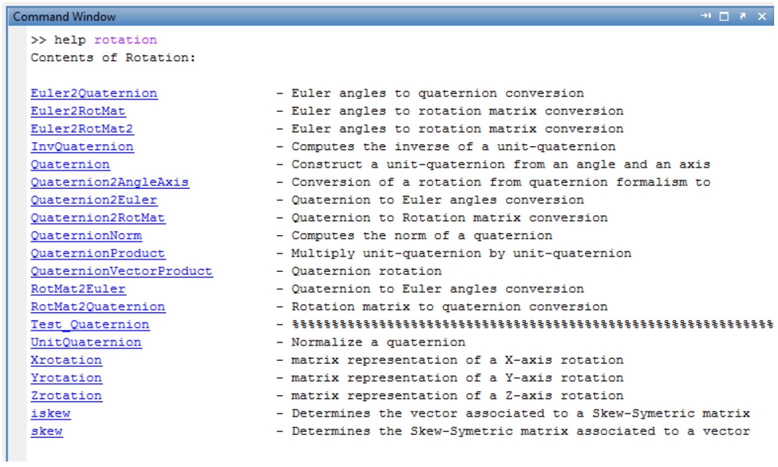
2.2.3 Other tools

The RT-MaG toolbox provides also set of functions to compute different representation of rotations (Euler angles, rotation matrix, quaternion). You can obtain a list of the available functions as their description (help menu) like this:

2.3 Create a model

In this part you can find how to build rapidly a model step by step with a lot of customization.

First you have to create a new blank Simulink model.



```
>> help rotation
Contents of Rotation:

Euler2Quaternion                  - Euler angles to quaternion conversion
Euler2RotMat                      - Euler angles to rotation matrix conversion
Euler2RotMat2                     - Euler angles to rotation matrix conversion
InvQuaternion                     - Computes the inverse of a unit-quaternion
Quaternion                        - Construct a unit-quaternion from an angle and an axis
Quaternion2AngleAxis              - Conversion of a rotation from quaternion formalism to
Quaternion2Euler                  - Quaternion to Euler angles conversion
Quaternion2RotMat                 - Quaternion to Rotation matrix conversion
QuaternionNorm                    - Computes the norm of a quaternion
QuaternionProduct                 - Multiply unit-quaternion by unit-quaternion
QuaternionVectorProduct           - Quaternion rotation
RotMat2Euler                      - Quaternion to Euler angles conversion
RotMat2Quaternion                 - Rotation matrix to quaternion conversion
Test_Quaternion                   - %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
UnitQuaternion                    - Normalize a quaternion
Xrotation                         - matrix representation of a X-axis rotation
Yrotation                         - matrix representation of a Y-axis rotation
Zrotation                         - matrix representation of a Z-axis rotation
iskew                             - Determines the vector associated to a Skew-Symetric matrix
skew                              - Determines the Skew-Symetric matrix associated to a vector
```

Figure 3: The list of the functions of the rotation tools provided by the RT-MaG toolbox.

2.3.1 Simulink configuration parameters

As the c-code of the Simulink model will be generated by Embedded coder, you need first configure the code generator:

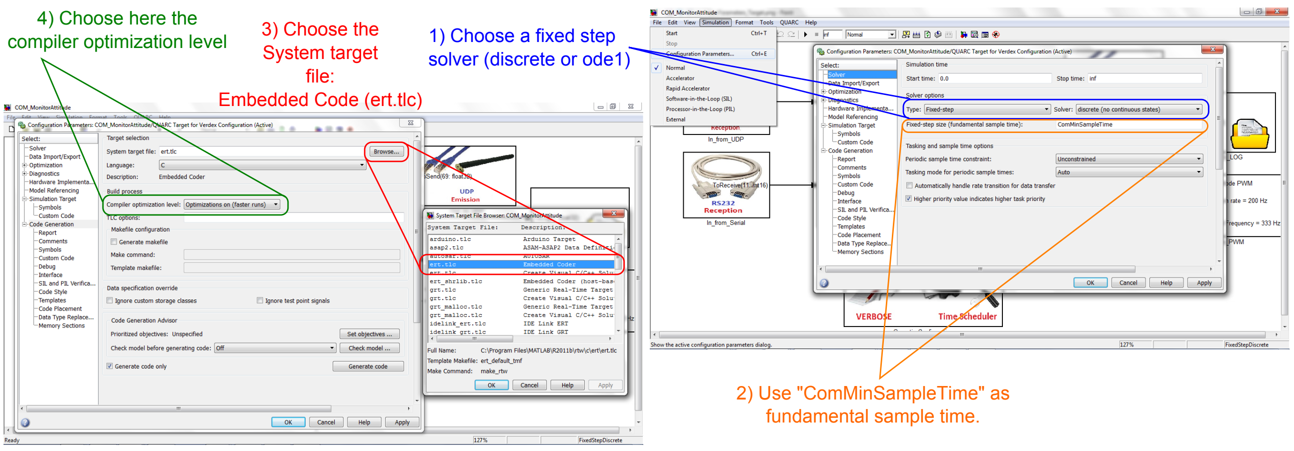

4) Choose here the compiler optimization level
3) Choose the System target file: Embedded Code (ert.tlc)
1) Choose a fixed step solver (discrete or ode1)
2) Use "ComMinSampleTime" as fundamental sample time.

Figure 4: How to configure your Simulink model in the Configuration Parameters

2.3.2 The I/Os blocks

If the RT-MaG tooblox has been correctly installed the RT-MaG toolbox, you can find all the I/Os block available in your Simulink browser: You just have to add the I/Os blocks you need for your application as

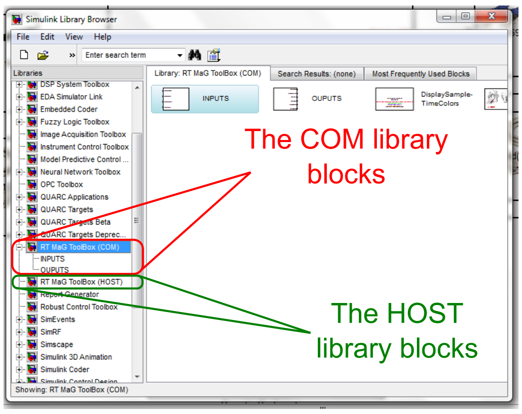


Figure 5: How to configure your Simulink model in the Configuration Parameters

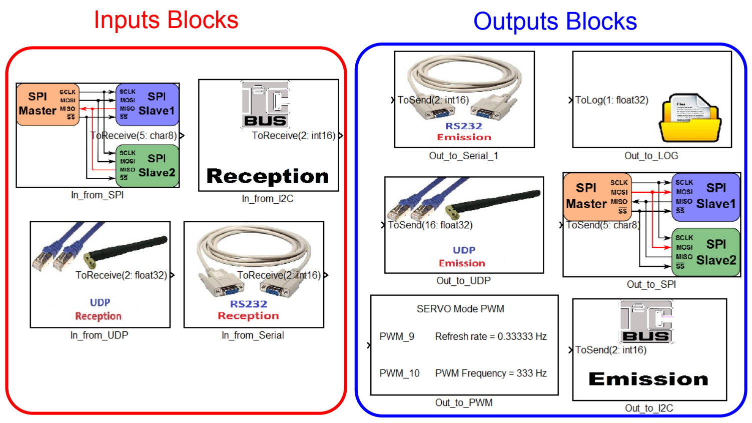
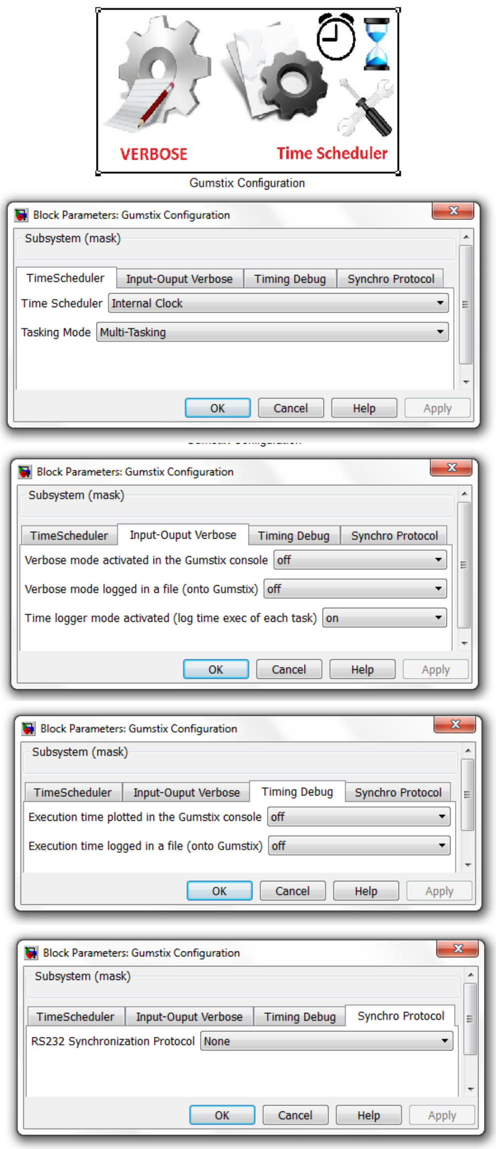


Figure 6: The inputs and Outputs block of the RT-MaG toolbox: SPI, I2C, File logging, PWM, RS232, UDP.

classic Simulink I/O blocks. You just have to respect the limitation of each I/Os, e.g., a emission and reception RS232 block need to use the same baudrate if they use the same device. The limitations of each blocks are described on the RT-MaG website in the Tutorials&Documentations section.

2.3.3 COM configuration block



Block Parameters: Gumstix Configuration
Subsystem (mask)
TimeScheduler | Input-Ouput Verbose | Timing Debug | Synchro Protocol

Time Scheduler: Internal Clock
Tasking Mode: Multi-Tasking

Verbose mode activated in the Gumstix console: off
Verbose mode logged in a file (onto Gumstix): off
Time logger mode activated (log time exec of each task): on

Execution time plotted in the Gumstix console: off
Execution time logged in a file (onto Gumstix): off

RS232 Synchronization Protocol: None

Figure 7: The COM configuration block of RT-MaG.

To configure properly your real-time application, the RT-MaG toolbox provides a configuration block allowing to choose in a large-wide of options:

- Time scheduler choice (timer, or I/O interrupt),
- tasking mode (multi task or single task application),
- I/O debug modes (via the console or via a file),
- Timing debug modes (via the console or via a file),
- Synchronization protocol

A detailed description of this block is given here on our website.

Each Simulink model destined to be run on a COM needs this block. There can have only one configuration block per model. You have to choose carefully all the options of this block because it manage how the application will be executed. The I/Os and timing debug modes are really useful to detect a misconfiguration but keep in mind they can slow down your application.

2.4 Generate the Standalone application

To generate the two models (Embedded and Host) you just built, you just have to run the following Matlab commands:

```
>> GenereHost(ModelName, HostSampleTime)
>> GenereCOM(ModelName, ComSampleTime)
```

As you can see, these two functions need you to specify a sample time, which corresponds to the minimum sample time of your model. The sample time you will pass as argument will be then assigned in your Matlab workspace as `HostMinSampleTime` and `ComMinSampleTime` for the host and the COM respectively. This is the variable given to the Simulink Configuration Parameters as the solver fixed-step sample time (fundamental sample time).

As a consequence if you use these two variables (`HostMinSampleTime` and `ComMinSampleTime`) in your model, you can easily generate the application for different sample time.

2.4.1 GenereCOM

As said previously, the generation of the sources needed to build the COM application is done by the command:

```
>> GenereCOM(ModelName, ComSampleTime)
```

The `GenereCOM` function evaluates all the mask of the different I/Os blocks present in your Simlink model and generates two structures in the workspace. These structures allow to check automatically that each I/O block are correctly used and they can work with the other I/O blocks. If their exist incompatibilities between different blocks, error messages will appear in the Matlab command window. You can display these two structure in the Matlab command window like this:

```
>>DispStruct(Gumstix)
    UDP(1):
        Emission(1):
            Name: Out_to_UDP
            NbData: 69
            DataType: float32
            IPaddress: 192.168.137.2
            Port: 2012
            Interface: wlan0
            SampleTime: 0.005
    RS232(1):
        Reception(1):
            Name: In_from_Serial
            NbData: 11
            DataType: int16
            PortIndex: 2
            Port: /dev/ttyO2
            BaudRate: 115200
            SampleTime: 0.005
            BlockingReception: 1
    Configuration(1):
        TimeScheduler: RS232 Reception
        ConsoleVerbose: off
        FileVerbose: off
        TimeLogger: on
        TimingDebug: off
        FileTimingDebug: off
        SynchroRS232(1):
            Type: Send-Wait
            TimeOut: 100
            Message: START
        TaskingMode: Multi-Tasking

>>DispStruct(IO_2_Process)
    UDP:
        Reception: 0
        Emission: 1
    RS232:
        Reception: 1
        Emission: 0
        PortData: {'/dev/ttyO2'   [2]   [115200]}
```

As you can see, these structures summarize all the I/O informations useful to generate the archive.

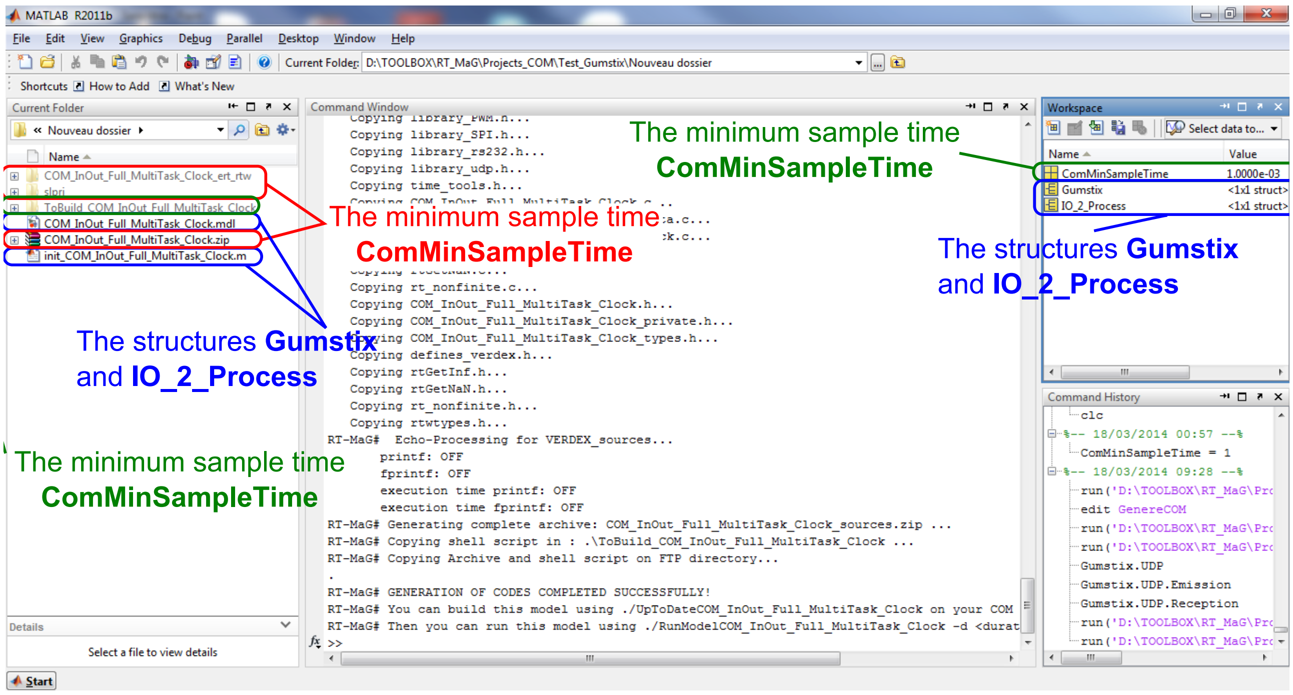
The `GenereCOM` function will also generate a lot of different files an directories. Some files and directories are generated by Embedded Coder itself and the other are generated by RT-MaG toolbox. Here you can find a rapid description of the main generated files and directories and their signification:

- directories:

- *slprj* is a directory generated by Simulink which contains some files used during the code generation. This folder is not useful after and can be deleted.
- *COM_**ModelName**_ert_rtw* is a directory generated by Embedded Coder, it contains all the c-files needed to declare the Simulink variables an realizes the model functions. Some of theses files were used by RT-MaG toolbox during the generation, but you can delete this folder if you want.
- *To_Build_COM_**ModelName*** is the directory generated by RT-MaG toolbox. It contains all the files needed to generate the standalone application on your COM (included the main part of the Embedded Coder files contained in the folder *COM_**ModelName***_ert_rtw). For a sake of simplicity an archive of this folder is available into it, and was also copied on your FTP server (*COM_**ModelName**_sources.zip*).

• files:

- In the folder *COM_**ModelName**_ert_rtw* their is `ert_main.c` which is a simple example (provided by Embedded Coder) of how to use and call the generated function in a standalone application. As you can see, the I/Os attribution, the interrupts, etc. are not done in this file, and that why the RT-MaG toolbox exists.
- *COM_**ModelName***`.zip` is an archive generated by Embedded Coder containing a lot of file but you does not need this. This file can be deleted.
- In the folder *To_Build_COM_**ModelName*** their is a file *main_COM_**ModelName**.c*. This file is the main file of the RT-MaG application.
- In the folder *To_Build_COM_**ModelName*** their is a file *genere_RT_COM_**ModelName***. This shell script allows to build the model on the COM and will be used by the shell script `UpToDate` (see section 2.5).


MATLAB R2011b
File Edit View Graphics Debug Parallel Desktop Window Help
Current Folder: D:\TOOLBOX\RT_MaG\Projects_COM\Test_Gumstix\Nouveau dossier
Shortcuts How to Add What's New
Current Folder
« Nouveau dossier ›
Name
COM_InOut_Full_MultiTask_Clock_ert_rtw
slprj
ToBuild_COM_InOut_Full_MultiTask_Clock
COM_InOut_Full_MultiTask_Clock.mdl
COM_InOut_Full_MultiTask_Clock.zip
init_COM_InOut_Full_MultiTask_Clock.m
Details
Select a file to view details
Start
Command Window
Workspace
Select data to...
Command History
The minimum sample time
ComMinSampleTime
The minimum sample time
ComMinSampleTime
The structures Gumstix
and IO_2_Process
The structures Gumstix
and IO_2_Process
The minimum sample time
ComMinSampleTime

```
   Copying library_PWM.h...
   Copying library_SPI.h...
   Copying library_rs232.h...
   Copying library_udp.h...
   Copying time_tools.h...
   Copying COM_InOut_Full_MultiTask_Clock.c...
                                           a.c...
                                           k.c...
   Copying rt_nonfinite.c...
   Copying COM_InOut_Full_MultiTask_Clock.h...
   Copying COM_InOut_Full_MultiTask_Clock_private.h...
   Copying COM_InOut_Full_MultiTask_Clock_types.h...
   Copying defines_verdex.h...
   Copying rtGetInf.h...
   Copying rtGetNaN.h...
   Copying rt_nonfinite.h...
   Copying rtwtypes.h...
RT-MaG#  Echo-Processing for VERDEX_sources...
       printf: OFF
       fprintf: OFF
       execution time printf: OFF
       execution time fprintf: OFF
RT-MaG# Generating complete archive: COM_InOut_Full_MultiTask_Clock_sources.zip ...
RT-MaG# Copying shell script in : .\ToBuild_COM_InOut_Full_MultiTask_Clock ...
RT-MaG# Copying Archive and shell script on FTP directory...
.
RT-MaG# GENERATION OF CODES COMPLETED SUCCESSFULLY!
RT-MaG# You can build this model using ./UpToDateCOM_InOut_Full_MultiTask_Clock on your COM
RT-MaG# Then you can run this model using ./RunModelCOM_InOut_Full_MultiTask_Clock -d <durat
>>
```

Name	Value
ComMinSampleTime	1.0000e-03
Gumstix	<1x1 struct>
IO_2_Process	<1x1 struct>

```
clc
%-- 18/03/2014 00:57 --%
ComMinSampleTime = 1
%-- 18/03/2014 09:28 --%
run('D:\TOOLBOX\RT_MaG\Pro
edit GenereCOM
run('D:\TOOLBOX\RT_MaG\Pro
run('D:\TOOLBOX\RT_MaG\Pro
Gumstix.UDP
Gumstix.UDP.Emission
Gumstix.UDP.Reception
run('D:\TOOLBOX\RT_MaG\Pro
run('D:\TOOLBOX\RT_MaG\Pro
```

Figure 8: The folders and the workspace variables obtain after a COM generation

Here is an example of the COM's model generation for the test model **InOut_Full_MultiTask_Clock**:

```
RT-MaG#  Remove previous build directories...
RT-MaG#       Try to remove: COM_InOut_Full_MultiTask_Clock_ert_rtw...
RT-MaG#            Done!
RT-MaG#       Try to remove: ToBuild_COM_InOut_Full_MultiTask_Clock...
RT-MaG#            Done!

RT-MaG# Generation of code for COM using Embedded Coder
    COM_InOut_Full_MultiTask_Clock...

### Starting build procedure for model: COM_InOut_Full_MultiTask_Clock
### Generating code into build folder:
    D:\TOOLBOX\RT_MaG\Projects_COM\Test_Gumstix\COM_InOut_Full_MultiTask_Clock_ert_rtw
### Invoking Target Language Compiler on COM_InOut_Full_MultiTask_Clock.rtw
### Using System Target File: C:\Program Files\MATLAB\R2011b\rtw\c\ert\ert.tlc
### Loading TLC function libraries
......
### Initial pass through model to cache user defined code
...
### Caching model source code
.........................................................................
### Writing header file COM_InOut_Full_MultiTask_Clock.h
.
### Writing header file COM_InOut_Full_MultiTask_Clock_types.h
### Writing header file rtwtypes.h
.
### Writing header file rtGetInf.h
### Writing source file rtGetInf.c
.
### Writing header file rtGetNaN.h
### Writing source file rtGetNaN.c
.
### Writing header file rt_nonfinite.h
### Writing source file rt_nonfinite.c
.
### Writing source file COM_InOut_Full_MultiTask_Clock.c
### Writing header file COM_InOut_Full_MultiTask_Clock_private.h
.
### Writing source file COM_InOut_Full_MultiTask_Clock_data.c
### Writing source file ert_main.c
.
### TLC code generation complete.
### Creating project marker file: rtw_proj.tmw
### Evaluating PostCodeGenCommand specified in the model
### Successful completion of code generation for model: COM_InOut_Full_MultiTask_Clock

RT-MaG# Begin to genere code for the StandAlone application using
    .\COM_InOut_Full_MultiTask_Clock_ert_rtw :

RT-MaG# Generating main file: main_COM_InOut_Full_MultiTask_Clock.c ...
RT-MaG#      Generating headers includes...
RT-MaG#      Generating global variables...
RT-MaG#      Generating I/O proper termination...
RT-MaG#      Generating Inputs attribution for Sample Time: 0.1 [s]...
RT-MaG#      Generating Outputs attribution for Sample Time: 0.1 [s]...
RT-MaG#      Generating Inputs attribution for Sample Time: 0.01 [s]...
RT-MaG#      Generating Outputs attribution for Sample Time: 0.01 [s]...
RT-MaG#      Generating Inputs attribution for Sample Time: 0.005 [s]...
RT-MaG#      Generating Outputs attribution for Sample Time: 0.005 [s]...
RT-MaG#      Generating Inputs attribution for Sample Time: 0.003 [s]...
RT-MaG#      Generating Outputs attribution for Sample Time: 0.003 [s]...
RT-MaG#      Generating Inputs attribution for Sample Time: 0.002 [s]...
RT-MaG#      Generating Outputs attribution for Sample Time: 0.002 [s]...
RT-MaG#      Generating Inputs attribution for Sample Time: 0.001 [s]...
RT-MaG#      Generating Outputs attribution for Sample Time: 0.001 [s]...
RT-MaG#      Generating I/O initialization...
RT-MaG#      Generating specific main's argument processing...
RT-MaG#  Generating defines file: defines_verdex.h ...
RT-MaG#  Generating shell script file: genere_RT_COM_InOut_Full_MultiTask_Clock ...
RT-MaG#  Generating complete build directory:  .\ToBuild_COM_InOut_Full_MultiTask_Clock
   Copying library_I2C.c...
   Copying library_LOG.c...
   Copying library_PWM.c...
   Copying library_SPI.c...
   Copying library_rs232.c...
   Copying library_udp.c...
   Copying time_tools.c...
```

```
    Copying library_I2C.h...
    Copying library_LOG.h...
    Copying library_PWM.h...
    Copying library_SPI.h...
    Copying library_rs232.h...
    Copying library_udp.h...
    Copying time_tools.h...
    Copying COM_InOut_Full_MultiTask_Clock.c...
    Copying COM_InOut_Full_MultiTask_Clock_data.c...
    Copying main_COM_InOut_Full_MultiTask_Clock.c...
    Copying rtGetInf.c...
    Copying rtGetNaN.c...
    Copying rt_nonfinite.c...
    Copying COM_InOut_Full_MultiTask_Clock.h...
    Copying COM_InOut_Full_MultiTask_Clock_private.h...
    Copying COM_InOut_Full_MultiTask_Clock_types.h...
    Copying defines_verdex.h...
    Copying rtGetInf.h...
    Copying rtGetNaN.h...
    Copying rt_nonfinite.h...
    Copying rtwtypes.h...
RT-MaG#  Echo-Processing for VERDEX_sources...
        printf: OFF
        fprintf: OFF
        execution time printf: OFF
        execution time fprintf: OFF
RT-MaG# Generating complete archive: COM_InOut_Full_MultiTask_Clock_sources.zip ...
RT-MaG# Copying shell script in : .\ToBuild_COM_InOut_Full_MultiTask_Clock ...
RT-MaG# Copying Archive and shell script on FTP directory...
.
RT-MaG# GENERATION OF CODES COMPLETED SUCCESSFULLY!
RT-MaG# You can build this model using ./UpToDateCOM_InOut_Full_MultiTask_Clock on
    your COM
RT-MaG# Then you can run this model using ./RunModelCOM_InOut_Full_MultiTask_Clock -d
    <duration> on your COM
```

2.4.2 GenereHost

As said previously, the generation of the real-time Simulink model for the host application is done by the command:

```
>> GenereHost(ModelName, HostSampleTime)
```

This command creates a **ModelName.rt-win64** application which will be used by the Quarc real-time target. Take a look to the section 2.5.2 to know how to use and run this model in real time.

2.5 Run your application

2.5.1 COM part

Now, you just need to download and compile the archive generated in the subsection 2.4.1. So, turn on your COM, log on, and go to the RT-MaG directory to build your application. If you use one of our RT-MaG Linux image, you can proceed as follow:

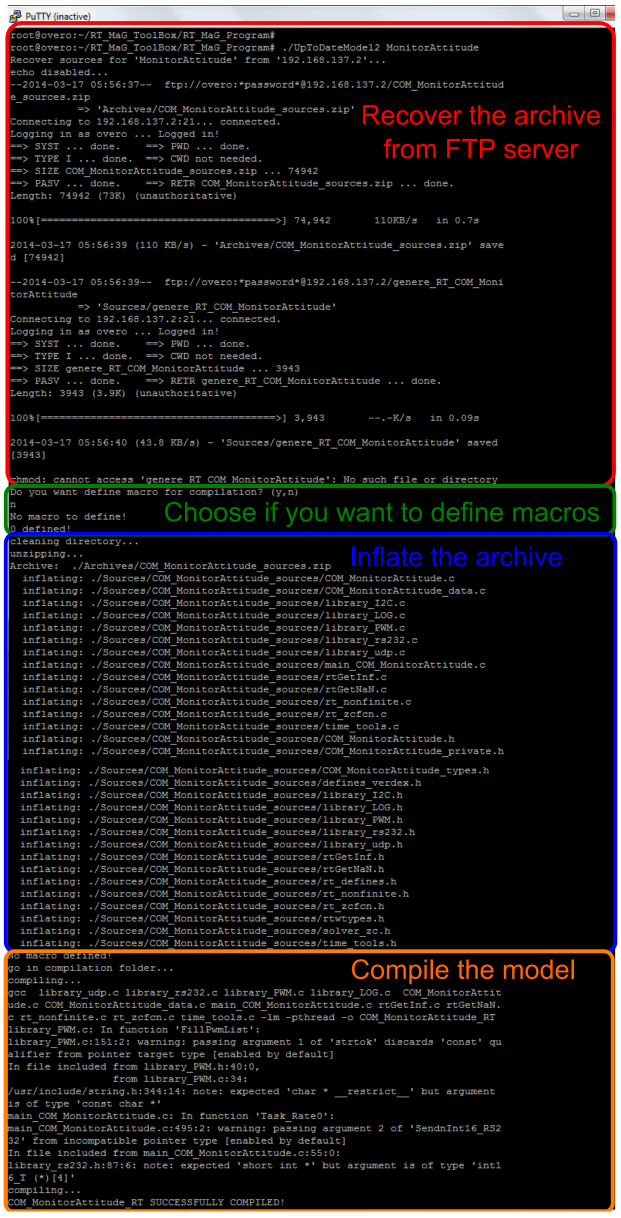
```
login:root
root@overo ~:cd RT_MaG_Toolbox/RT_MaG_Program
```

Then, the archive can be automatically downloaded and compiled on the COM using the Linux command:

```
root@overo ~:./UpToDateModel ModelName
```

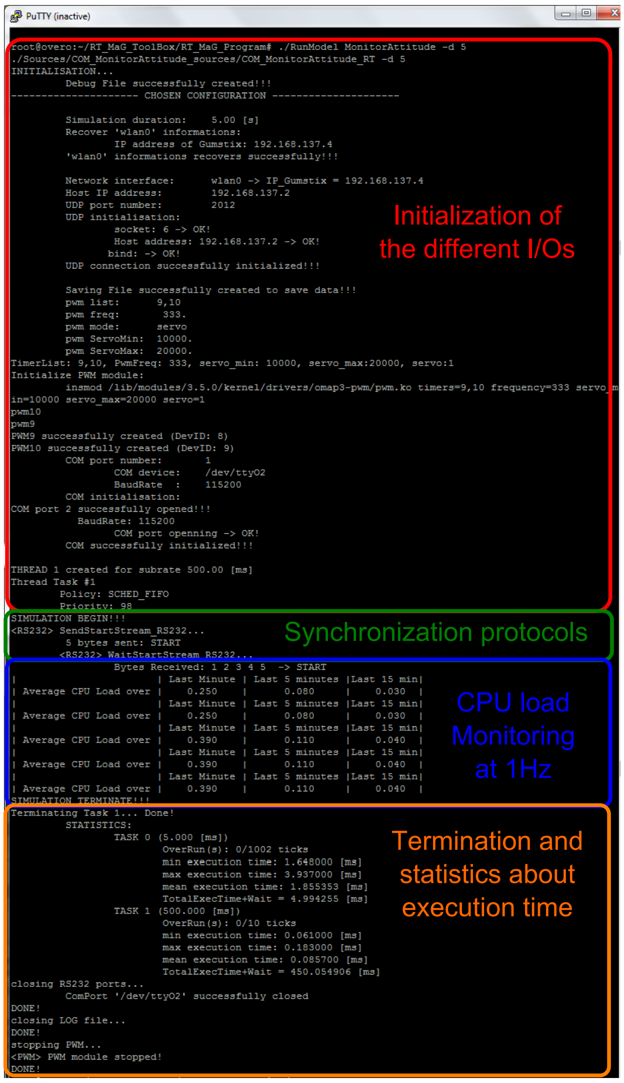
You will obtain something like the figure 9 for the compilation. The archive is downloaded, then you can choose to define some macros (to activate some debug mode even if their are not activated on the Simulink model). Finally, the archive is inflated and the program is compiled.

Then the real-time application could be executed with the command:


PuTTY (inactive)
Recover the archive from FTP server
Choose if you want to define macros
Inflate the archive
Compile the model

```
root@overo:~/RT_MaG_ToolBox/RT_MaG_Program#
root@overo:~/RT_MaG_ToolBox/RT_MaG_Program# ./UpToDateModel2 MonitorAttitude
Recover sources for 'MonitorAttitude' from '192.168.137.2'...
echo disabled...
--2014-03-17 05:56:37--  ftp://overo:*password*@192.168.137.2/COM_MonitorAttitud
e_sources.zip
           => 'Archives/COM_MonitorAttitude_sources.zip'
Connecting to 192.168.137.2:21... connected.
Logging in as overo ... Logged in!
==> SYST ... done.    ==> PWD ... done.
==> TYPE I ... done.  ==> CWD not needed.
==> SIZE COM_MonitorAttitude_sources.zip ... 74942
==> PASV ... done.    ==> RETR COM_MonitorAttitude_sources.zip ... done.
Length: 74942 (73K) (unauthoritative)

100%[======================================>] 74,942       110KB/s   in 0.7s

2014-03-17 05:56:39 (110 KB/s) - 'Archives/COM_MonitorAttitude_sources.zip' save
d [74942]

--2014-03-17 05:56:39--  ftp://overo:*password*@192.168.137.2/genere_RT_COM_Moni
torAttitude
           => 'Sources/genere_RT_COM_MonitorAttitude'
Connecting to 192.168.137.2:21... connected.
Logging in as overo ... Logged in!
==> SYST ... done.    ==> PWD ... done.
==> TYPE I ... done.  ==> CWD not needed.
==> SIZE genere_RT_COM_MonitorAttitude ... 3943
==> PASV ... done.    ==> RETR genere_RT_COM_MonitorAttitude ... done.
Length: 3943 (3.9K) (unauthoritative)

100%[======================================>] 3,943       --.-K/s   in 0.09s

2014-03-17 05:56:40 (43.8 KB/s) - 'Sources/genere_RT_COM_MonitorAttitude' saved
[3943]

chmod: cannot access 'genere_RT_COM_MonitorAttitude': No such file or directory
Do you want define macro for compilation? (y,n)
n
No macro to define!
0 defined!
cleaning directory...
unzipping...
Archive:  ./Archives/COM_MonitorAttitude_sources.zip
  inflating: ./Sources/COM_MonitorAttitude_sources/COM_MonitorAttitude.c
  inflating: ./Sources/COM_MonitorAttitude_sources/COM_MonitorAttitude_data.c
  inflating: ./Sources/COM_MonitorAttitude_sources/library_I2C.c
  inflating: ./Sources/COM_MonitorAttitude_sources/library_LOG.c
  inflating: ./Sources/COM_MonitorAttitude_sources/library_PWM.c
  inflating: ./Sources/COM_MonitorAttitude_sources/library_rs232.c
  inflating: ./Sources/COM_MonitorAttitude_sources/library_udp.c
  inflating: ./Sources/COM_MonitorAttitude_sources/main_COM_MonitorAttitude.c
  inflating: ./Sources/COM_MonitorAttitude_sources/rtGetInf.c
  inflating: ./Sources/COM_MonitorAttitude_sources/rtGetNaN.c
  inflating: ./Sources/COM_MonitorAttitude_sources/rt_nonfinite.c
  inflating: ./Sources/COM_MonitorAttitude_sources/rt_zcfcn.c
  inflating: ./Sources/COM_MonitorAttitude_sources/time_tools.c
  inflating: ./Sources/COM_MonitorAttitude_sources/COM_MonitorAttitude.h
  inflating: ./Sources/COM_MonitorAttitude_sources/COM_MonitorAttitude_private.h

  inflating: ./Sources/COM_MonitorAttitude_sources/COM_MonitorAttitude_types.h
  inflating: ./Sources/COM_MonitorAttitude_sources/defines_verdex.h
  inflating: ./Sources/COM_MonitorAttitude_sources/library_I2C.h
  inflating: ./Sources/COM_MonitorAttitude_sources/library_LOG.h
  inflating: ./Sources/COM_MonitorAttitude_sources/library_PWM.h
  inflating: ./Sources/COM_MonitorAttitude_sources/library_rs232.h
  inflating: ./Sources/COM_MonitorAttitude_sources/library_udp.h
  inflating: ./Sources/COM_MonitorAttitude_sources/rtGetInf.h
  inflating: ./Sources/COM_MonitorAttitude_sources/rtGetNaN.h
  inflating: ./Sources/COM_MonitorAttitude_sources/rt_defines.h
  inflating: ./Sources/COM_MonitorAttitude_sources/rt_nonfinite.h
  inflating: ./Sources/COM_MonitorAttitude_sources/rt_zcfcn.h
  inflating: ./Sources/COM_MonitorAttitude_sources/rtwtypes.h
  inflating: ./Sources/COM_MonitorAttitude_sources/solver_zc.h
  inflating: ./Sources/COM_MonitorAttitude_sources/time_tools.h
No macro defined!
go in compilation folder...
compiling...
gcc  library_udp.c library_rs232.c library_PWM.c library_LOG.c  COM_MonitorAttit
ude.c COM_MonitorAttitude_data.c main_COM_MonitorAttitude.c rtGetInf.c rtGetNaN.
c rt_nonfinite.c rt_zcfcn.c time_tools.c -lm -pthread -o COM_MonitorAttitude_RT
library_PWM.c: In function 'FillPwmList':
library_PWM.c:151:2: warning: passing argument 1 of 'strtok' discards 'const' qu
alifier from pointer target type [enabled by default]
In file included from library_PWM.h:40:0,
                 from library_PWM.c:34:
/usr/include/string.h:344:14: note: expected 'char * __restrict__' but argument
is of type 'const char *'
main_COM_MonitorAttitude.c: In function 'Task_Rate0':
main_COM_MonitorAttitude.c:495:2: warning: passing argument 2 of 'SendnInt16_RS2
32' from incompatible pointer type [enabled by default]
In file included from main_COM_MonitorAttitude.c:55:0:
library_rs232.h:87:6: note: expected 'short int *' but argument is of type 'int1
6_T (*)[4]'
compiling...
COM_MonitorAttitude_RT SUCCESSFULLY COMPILED!
```

Figure 9: Results of the `UpToDateModel` command.


PuTTY (inactive)
Initialization of the different I/Os
Synchronization protocols
CPU load Monitoring at 1Hz
Termination and statistics about execution time

```
root@overo:~/RT_MaG_ToolBox/RT_MaG_Program# ./RunModel MonitorAttitude -d 5
./Sources/COM_MonitorAttitude_sources/COM_MonitorAttitude_RT -d 5
INITIALISATION...
        Debug File successfully created!!!
--------------------- CHOSEN CONFIGURATION ---------------------

        Simulation duration:    5.00 [s]
        Recover 'wlan0' informations:
                IP address of Gumstix: 192.168.137.4
        'wlan0' informations recovers successfully!!!

        Network interface:      wlan0 -> IP_Gumstix = 192.168.137.4
        Host IP address:        192.168.137.2
        UDP port number:        2012
        UDP initialisation:
                socket: 6 -> OK!
                Host address: 192.168.137.2 -> OK!
               bind: -> OK!
        UDP connection successfully initialized!!!

        Saving File successfully created to save data!!!
        pwm list:       9,10
        pwm freq:        333.
        pwm mode:       servo
        pwm ServoMin:   10000.
        pwm ServoMax:   20000.
TimerList: 9,10, PwmFreq: 333, servo_min: 10000, servo_max:20000, servo:1
Initialize PWM module:
        insmod /lib/modules/3.5.0/kernel/drivers/omap3-pwm/pwm.ko timers=9,10 frequency=333 servo_m
in=10000 servo_max=20000 servo=1
pwm10
pwm9
PWM9 successfully created (DevID: 8)
PWM10 successfully created (DevID: 9)
        COM port number:        1
                COM device:     /dev/ttyO2
                BaudRate  :     115200
        COM initialisation:
COM port 2 successfully opened!!!
          BaudRate: 115200
                COM port openning -> OK!
        COM successfully initialized!!!

THREAD 1 created for subrate 500.00 [ms]
Thread Task #1
        Policy: SCHED_FIFO
        Priority: 98
SIMULATION BEGIN!!!
<RS232> SendStartStream_RS232...
        5 bytes sent: START
        <RS232> WaitStartStream_RS232...
                Bytes Received: 1 2 3 4 5  -> START
|                       | Last Minute | Last 5 minutes |Last 15 min|
| Average CPU Load over |    0.250    |      0.080     |   0.030   |
|                       | Last Minute | Last 5 minutes |Last 15 min|
| Average CPU Load over |    0.250    |      0.080     |   0.030   |
|                       | Last Minute | Last 5 minutes |Last 15 min|
| Average CPU Load over |    0.390    |      0.110     |   0.040   |
|                       | Last Minute | Last 5 minutes |Last 15 min|
| Average CPU Load over |    0.390    |      0.110     |   0.040   |
|                       | Last Minute | Last 5 minutes |Last 15 min|
| Average CPU Load over |    0.390    |      0.110     |   0.040   |
SIMULATION TERMINATE!!!
Terminating Task 1... Done!
        STATISTICS:
                TASK 0 (5.000 [ms])
                        OverRun(s): 0/1002 ticks
                        min execution time: 1.648000 [ms]
                        max execution time: 3.937000 [ms]
                        mean execution time: 1.855353 [ms]
                        TotalExecTime+Wait = 4.994255 [ms]
                TASK 1 (500.000 [ms])
                        OverRun(s): 0/10 ticks
                        min execution time: 0.061000 [ms]
                        max execution time: 0.183000 [ms]
                        mean execution time: 0.085700 [ms]
                        TotalExecTime+Wait = 450.054906 [ms]
closing RS232 ports...
        ComPort '/dev/ttyO2' successfully closed
DONE!
closing LOG file...
DONE!
stopping PWM...
<PWM> PWM module stopped!
DONE!
```

Figure 10: Results of the `RunModel` command for a duration of 5 seconds.

```
root@overo~:./RunModel ModelName -d MyDuration
```

where `MyDuration` is the time during which the application has to be executed. You can find an example of a program execution in figure 10.

After the inputs and outputs initialization, the synchronization protocols are executed on the RS232 device to check that the remote device is ready (optional). Then the program begins and the CPU load is plotted each second. In addition, if overruns occur for some task their are plotted immediately. When the program finishes, it displays some statistics about execution times and eventual overruns and it frees the devices used by the program.

Remark 2: During an execution, you can type CTRL+C to quit immediately and properly the program, i.e., the program will free all the resources and quit.

2.5.2 Host part

Once the **ModelName.rt-win64** was generated for your host model, you just have to load it on the real-time Quarc target and then to play it. During the execution of the model, you will be able to modify all the parameter you want in the host model. If you send this parameter to your COM model via UDP, you will be able to tune your controller parameters running on the COM model. This is the ideal way to achieve rapid prototyping of controllers in real-time.

To run this model, you need first to load it on the real-time target which will execute this model in real-time. To do this just click on "Connect-to-target" as described by the figure 11. Once the target is connected, you can

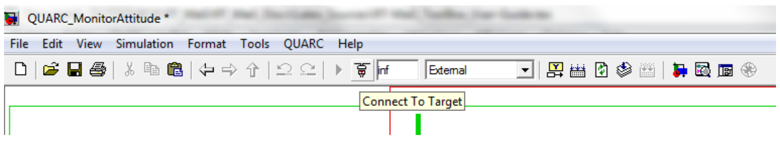


Figure 11: The connect to target step for loading the model on the real-time target before running it.

run the model by simply click on "Start real-time code" as described by figure 12. Then during the simulation

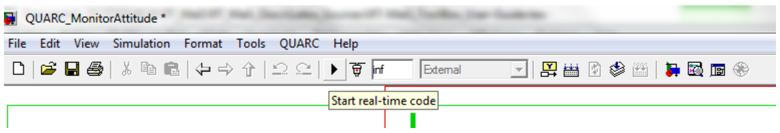


Figure 12: The start real-time code step for running a real-time Simulink model on the host computer.

you can change every parameters you want and it will take effect immediately.

3 Advanced functionalities

3.1 The debug modes

You can activate in the configuration block described in section 2.3.3 different debug modes. These different mode allow to detect eventual I/O failure due to misconfiguration, hardware failure or missing remote device. You have three kind of debug modes:

- **Input-Output Verbose:** When activated this mode reports all the Inputs and Outputs results, i.e., the number of received data, their values, and the eventual failure (unavailable resource, etc.). The results can be plotted in real-time in the console or just written in a .debug file.
- **Timing Debug:** When activated this mode reports the execution time of each task, i.e., the duration of each I/O access, the duration of each task. The results can be plotted in real-time in the console or just written in a .debug file.
- **Time Logger:** When activated this mode reports the execution time of each task in a file. The only difference between the previous debug mode is it reports the time of a complete task (Inputs of the task + the task + Outputs of the task).

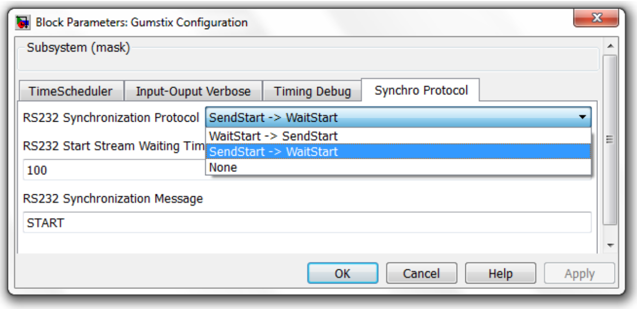
Remark 3: You can also activate these debug mode by defining macros during the compilation of the application on your COM. This is an interesting thing because you are not obliged to generate the archive again. At the question "Do you want define macro for the compilation?", respond "y" and the define one or several of this macro, separated buy a space:

- ECHO_PRINTF: to activate the inputs-outputs verbose mode in a console,
- ECHO_FPRINTF: to activate the verbose mode in a file,
- ECHO_PRINTF_TIME: to activate the timing debug mode in a console,
- ECHO_FPRINTF_TIME: to activate the timing debug mode in a file,

3.2 The synchronization protocols

For the RS232 connection, you can choose to activate or not a synchronization protocol. As it is not possible in a RS232 bus to separate frame, you can choose to activate a synchronization protocol. This protocol consists in a simple stream, send at the beginning of the program allowing to say to the remote device that the program begins. This synchronization protocol consists in a Start stream which is send to the remote device, and the program starts when the remote device sends the same start stream in response.

Remark 4: If a synchronization protocol fails, the program will not begin and you can see what fail. For example, which start stream was received back. A synchronization protocol fail often occur if the baudrate of your RS232 device and the remote device are not the same.



Block Parameters: Gumstix Configuration

Subsystem (mask)

TimeScheduler | Input-Ouput Verbose | Timing Debug | Synchro Protocol

RS232 Synchronization Protocol: SendStart -> WaitStart

WaitStart -> SendStart
SendStart -> WaitStart
None

RS232 Start Stream Waiting Tim

100

RS232 Synchronization Message

START

Gumstix Configuration